Object-Oriented Design

Basics
Design Themes

1

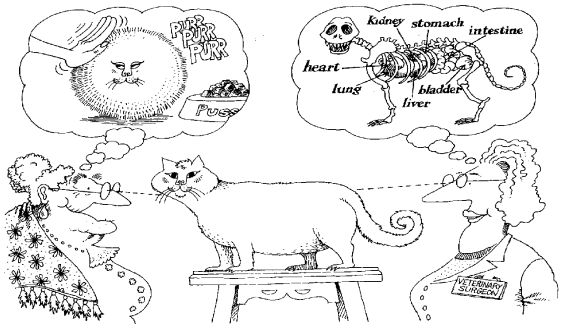
Basic Knowledge

Design themes:

- ✓Abstraction
- ✓Encapsulation
- ✓Delegation
- ✓Modularity
- ✓Hierarchy
- ✓Typing

2

Design theme: Abstraction

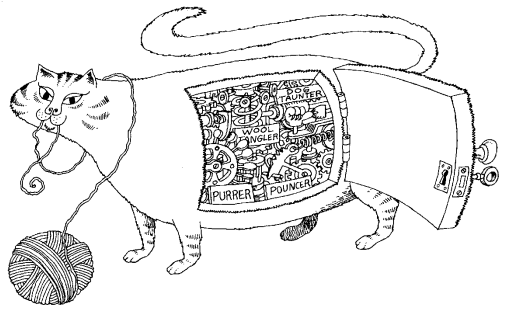


from Object oriented design, Grady Booch, 1991

- An abstraction points at the essential features of an object, depending on the point of view of the observer.

3

Design theme: Encapsulation



- Encapsulation hides the implementation details of an object.
- The user of an object only is confronted with the outside of an object, that is its behaviour, not the source of that behaviour.
- In the design, the abstraction of an object must precede any decision about the implementation.

4

Design theme: Encapsulation

- *Encapsulation* protects data from being set in an improper way. With *encapsulated* data, any calculations or checks that the class does on the data are preserved, since the data can't be accessed directly.

  → So Encapsulation does more than just hide information; it makes sure the methods you write to work with your data are actually used!

- Encapsulation separates your data from your app's behavior. Then you can control how each part is used by the rest of your application.

Design theme: Delegation

- What?
  Delegation is when an object needs to perform a certain task, and instead of doing that task directly, it asks another object to handle the task (or sometimes just a part of the task).

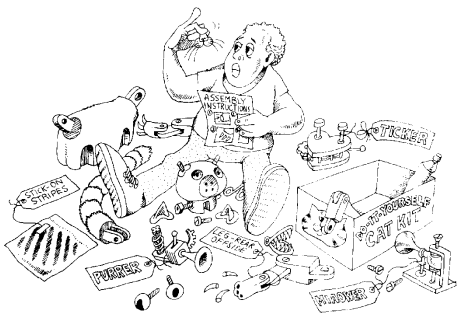
- Advantage?
  Delegation makes your code more reusable.

  Delegation lets each object worry about his task. This means your objects are more INDEPENDENT of each other, or more LOOSELY COUPLED.

Design theme: Delegation

- Loosely coupled is when the objects in your application each have a specific job to do, and they do only this job.
- Advantage?
  - Loosely coupled objects can be taken from one app and easily reused in another, because they're not tightly tied to other objects'code.
  - Loosely coupled applications are usually more flexible, and easy to change.

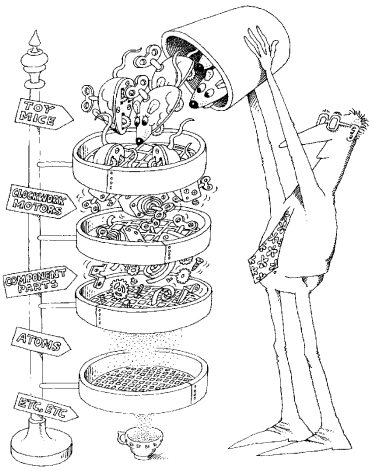
Design theme: Modularity



- When building complex systems it comes in handy to use parts (components, modules).
- A module, like a class, hides (encapsulates) the implementation details. The user is only confronted with the interface of the part (component, module).

8

Design theme: Hierarchy



- A hierarchical set of abstractions it helps to get a better understanding of the domain and to reduce the complexity of the problem.
  - Class-hierarchy ("kind of" relationship) that uses inheritance
  - Object-hierarchy ("part of" relationship)

9

Design theme: Hierarchy

Exercise:

- Make an object hierarchy (part of relationship) for
  - a car
- Make a class hierarchy (kind of relationship) using the following concepts (you are allowed to add more concepts in order to build the hierarchy):
  - chair, table, couch, cupboard, coffee table, armchair, kitchen cupboard, wardrobe, bed

10

Design theme: Typing

- Typing enforces the class of an object, such that objects of different type cannot be interchanged.
- Programming languages can be strongly typed (Ada, C++) or, at the other extreme, untyped (Smalltalk, Lisp).
- Programming languages that have strong typing have more control mechanisms than untyped languages, that offer more flexibility.
- Static or dynamic binding refers to the the moment that the type of variables and expressions is established.
  - Static: at the time of compilation
  - Dynamic: during execution
- example:

When a program uses a list that can be sorted and it is only at execution time that we get to know what will be sorted (words, numbers, students, …), then the correct sorting method can only be chosen when the exact object is known., dynamic, at execution time.

The fact that **one name** refers to **different methods** is called **polymorphism**